Notes

USF Teaching Survey: LINK . Please fill out! Today during lab time?

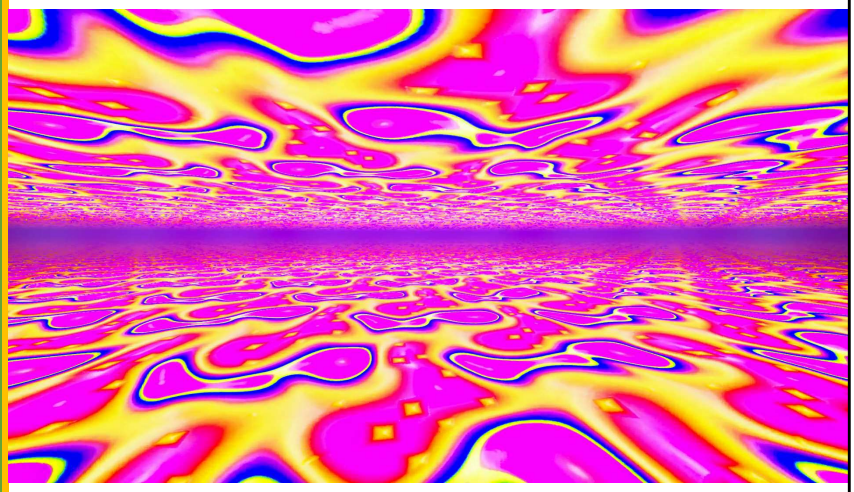
Project02 due Friday! You should be getting help if you need it.

Final exam Monday Dec 12, 10am-noon. See me if you have a conflict.

CS112 –
Java
Programming

Fall 2022

Graphics



Graphics

We pick a computer language because it is a good fit for our current task

- Available libraries a big part of choice

Java has the best graphics library I have seen

We do not have enough time to cover graphics in Java completely.
DO have enough time to learn some useful TOOLS, and some useful PARADIGMS

SHOW: Fishies, AndrewProto01

Why learn about Graphics?

Fun and useful

Learn new programming paradigms

Multi Threads

Event Handling

Intro to Java graphics libraries

- Abstract Windowing Toolkit "AWT"
- Swing
- JavaFX

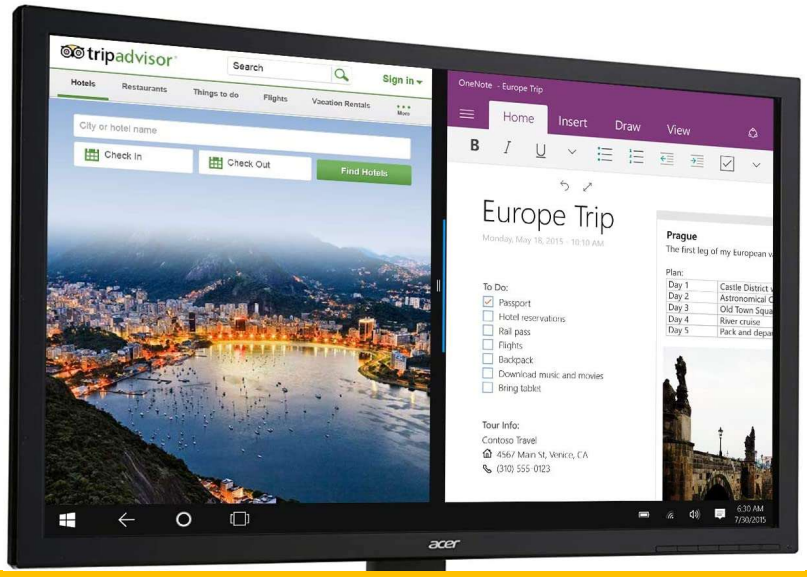
Textbook *Java Software Solutions*
- Version 8 and Version 9 differ in graphics coverage

We will work with Swing

AWT: original Java graphics pkg, supports basic components such as color
Swing: much more powerful, easier to use. Partly based on AWT
JavaFX: new paradigm, I have not seen much use yet myself. Intro'd in 2008 and still has not replaced Swing.
But Swing is "deprecated", meaning "Oracle hopes you will use JavaFX"

Growth in use of HTML5 for graphics, with huge growth in web/browser based applications.

Swing Basics



Entire graphics screen is represented with a **JFrame**. Not visible, a top-level manager.
Individual windows on the screen are represented with **JPanel**s, or some class derived from **JPanel**.
A **JPanel** can contain other **JPanel**s, recursively.

Basic Swing Graphics Program

```
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JPanel;

static public void main(String[] args) {
 JFrame jf = new JFrame("Name of Application");
 jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 JPanel jp = new JPanel();
 jp.setPreferredSize(new Dimension(700, 500));

 jf.add(jp); // insert JPanel into JFrame
 jf.pack(); // set size of top-level window
 jf.setVisible(true);
}
```

Go ahead and type along
PAUL EXPLAIN EACH LINE

Cool Programming Concept #1:
Multithreading

Why is our program still running?

And what is a "Thread"?

Java Graphics programs automatically "spawn" a second thread

- `jf.setVisible();`
- `setDefaultCloseOperation()…`

Thread = "independent path or flow of execution"
Programs can have >1 thread. Can create and destroy threads.
Computer CPU can run (usually) 4 threads at the same time. Expensive computers can do more. Most expensive compr my group @ work ever used had 256.
OS can run thousands of threads in parallel, by switching between them rapidly. "Thread switching", "context switching".

Can Multithreading make a program simpler?

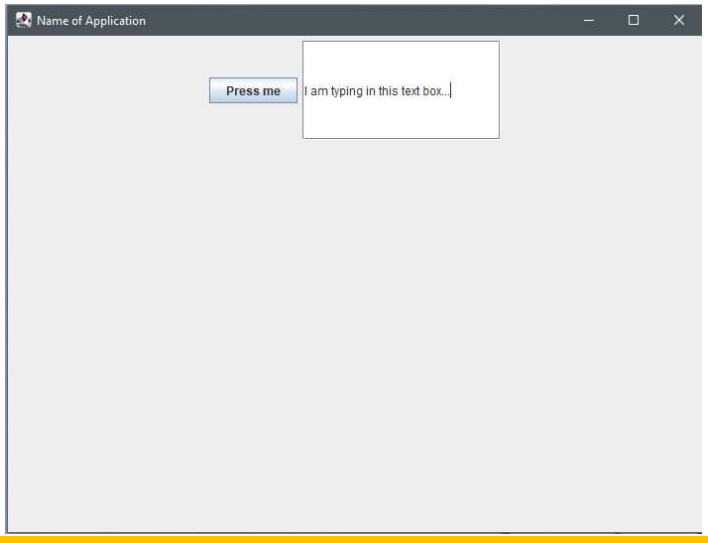
For example

- One thread prompts user for input and forwards it to another thread for processing. Separate threads ensure program is responsive to the user, without "freezing"
- Multi threads do expensive processing in parallel, to finish faster: **Mandelbrot.java**

CS 220, CS 272 spent much more time on multithreaded programming

Cool Programming Concept #2:
Event Handling

Let's have our program do something



ADD a JButton, JTextField

Well, we have more stuff on the screen. Looks like it might be useful. But doesn't do anything yet.

Events

Lots of events we might want our program to handle:

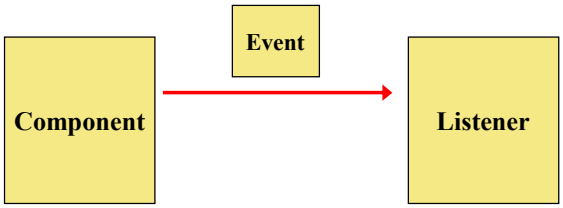
- User presses a button or selects a menu entry
- User enters text in a text box
- User clicks in a window
- User resizes a window
- User moves the mouse
- Timer expires
- Input data arrives on a socket

EVENT HANDLING PARADIGM:

- don't program or even know how program execution will flow
- Write handlers for events we want to handle
- Tell Java how to tie handlers ("listen", "register event handlers") to those events
- When program is run, the events trigger the handlers: Java system calls handlers when events detected

EXTEND THE PROGRAM TO DO SOMETHING!!

Events and Listeners



A component object generates an event

A corresponding listener object is designed to respond to the event

When the event occurs, the component calls the appropriate method of the listener, passing an object that describes the event

More events:
Timers

Timer class

Timer calls an event handler periodically.

```
Timer myTimer = new Timer(PeriodInMilliseconds, EventHandler);
myTimer.start();
```

If we add `Timer` to our previous code, how does `actionPerformed()` know whether an event came from `JButton` or `Timer`?

```
myTimer.setActionCommand("Timer");
jb.setActionCommand("PressMe");
```

Timer does not need separate call to set event handler. Must be part of constructor.

actionListener() can look at the Event argument to see who sent the event.
ADD IT TO CODE!

One more thing:
Colors

Colors in Swing

Swing components (`JPanels, JButtons`, etc) have a method to set background color.

```
setBackground(Color);
```

`Color` is a class in AWT that represents a color. Several are predefined as constants:

- `Color.BLACK`
- `Color.BLUE`
- `Color.RED`
- `Color.GREEN`
- `Color.YELLOW`
- `etc`

Or you can create a color by specifying Red, Green, and Blue values (0 -> 255).

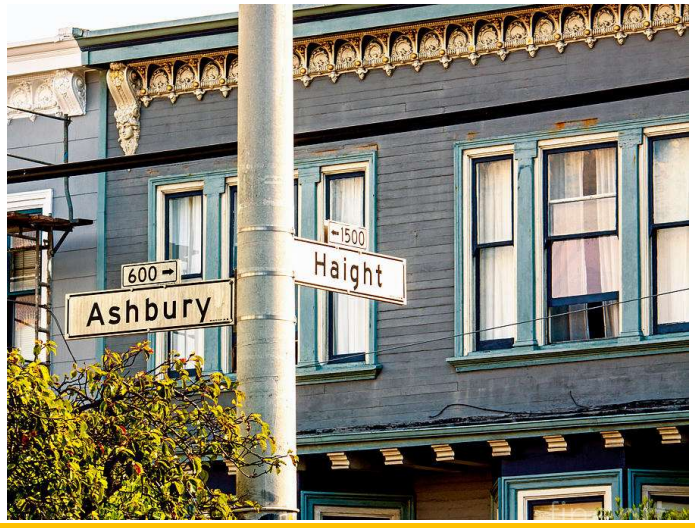
```
Color gold = new Color(240, 200, 40);
```

ADD THIS TO THE CODE!

The human eye has 3 color sensors, called "cones" in your retina. Some animals have fewer (none), some have more. Humans cannot distinguish between pure color (e.g. gold) and a suitably balanced mix of red/green/blue. TV's pixels take advantage of that: don't have millions of color elements, only 3.

Final Homework!

Your own LavaLamp



Anyone attending university 6 blocks from Haight-Ashbury ought to have their own Lava Lamp.

Shouldn't be too hard. I gave you the code we just developed. You develop your own LavaLamp program.
Not blobs of color but gradually changing solid background color.
Secret is to make small but random changes to the background color.